# Lab 1 – Introduction to R

August 13 & 14, 2018
FANR 6750

Richard Chandler and Bob Cooper
University of Georgia

## Today's Topics



## Good and Not So Good Things About R

Good

- Powerful platform for statistical analysis
- Many packages written for ecologists
- It's free
- Scripts save time
- **R** teaches you statistics

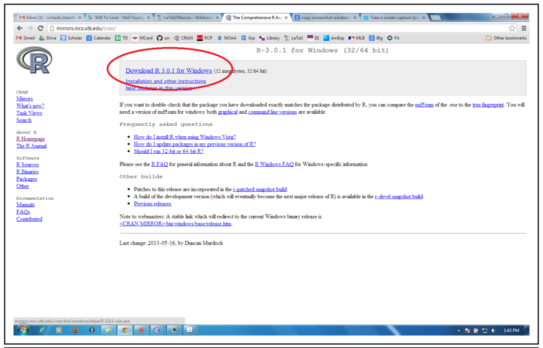
Not so good??

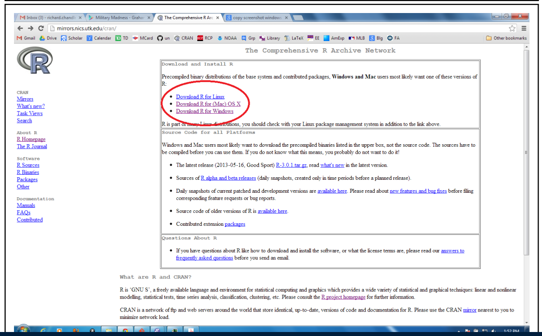
- Steep learning curve
- Help pages written for people familiar with **R**
- Developed by statisticians for statisticians
- Not as fast as some languages



## Downloading R

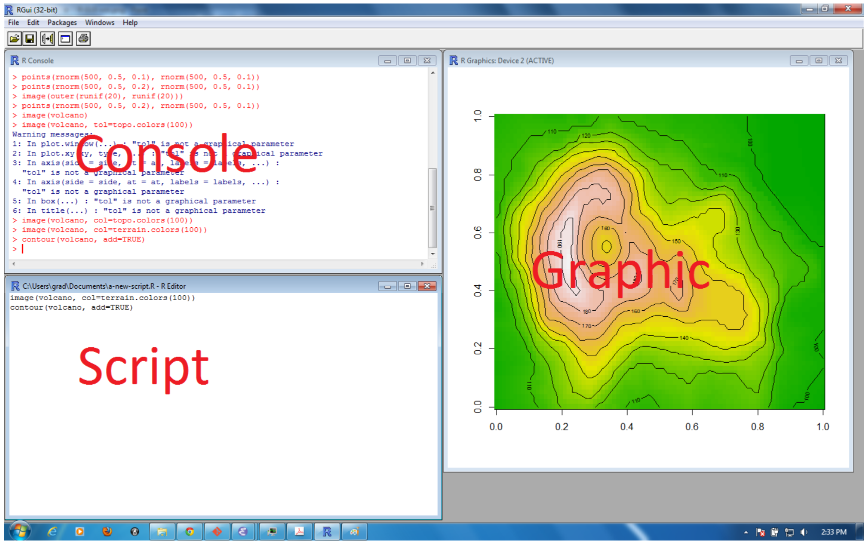
- Go to `www.r-project.org`
- Click on "CRAN"
- Choose a mirror near you (there is one in TN)





# Using the R GUI

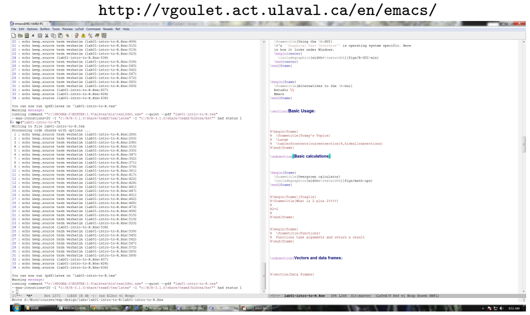
**R**'s "Graphical User Interface" is operating system specific. Here is how it looks under Windows.



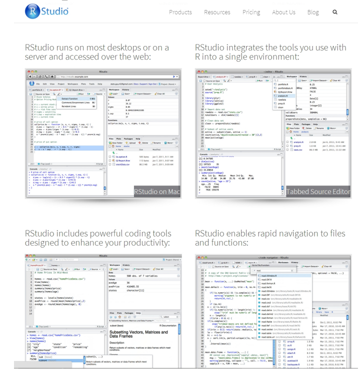



# Alternatives to the R Gui

Emacs and ESS
`http://vgoulet.act.ulaval.ca/en/emacs/`



RStudio
`http://www.rstudio.com/`



You are encouraged to learn and use these programs, but we will not use them for instruction because we want to focus on **R** itself, not the interface.



# How to read the code in the lab slides

Anything in a shaded box like this one is **R** code:

```
2+2

## [1] 4
```

The line `2+2` is **R** code (input). You can copy and paste it into the console. Note that the command prompt (>) is not shown.

The line `## [1] 4` is output. Output is always indicated by two hash signs (##). Anything after a # is ignored by **R** at the command line.

You can copy and paste the entire code box directly into your console, but it might be easier to work with the **R** script that accompanies the PDF: `lab-intro-to-R.R`.



# Overgrown calculator

Square-root of 3

```
sqrt(3)

## [1] 1.732051
```

6 squared

```
6^2

## [1] 36
```

cosine of $\pi$

```
cos(pi)

## [1] -1
```

# Objects and assignment arrow

Everything in **R** is an object. We can create objects using the `<-` assignment arrow.

In this example, we assign the value 2 to the object `y`:

```
y <- 2
```

You cannot have a space between `<` and `-`

You can use `=` instead of `<-` but this can cause confusion

Typing the name of an object returns its value:

```
y

## [1] 2
```



# Vectors

We store data in objects so that they can be easily manipulated:

```
y*2+1

## [1] 5
```

Usually, we want more than one number in an object. In statistics, a vector is simply a set of numbers that can be thought of as a row or column of a matrix

The easiest way to create a vector is to use the `c` function to "combine" numbers:

```
z <- c(-1, 9, 33, -4)
z

## [1] -1  9 33 -4
```



# Other useful ways of creating vectors

A sequence of numbers

```
x1 <- 1:3 # Same as: x1 <- c(1, 2, 3)
          # Note: anything after "#" is a comment
x1

## [1] 1 2 3
```

`seq` is more general

```
x2 <- seq(from=1, to=7, by=2)
x2

## [1] 1 3 5 7
```

Use `rep` to repeat elements of a vector

```
rep(x2, times=2)

## [1] 1 3 5 7 1 3 5 7
```



# Help with a function

These do the same thing

```
?rep
help(rep)
```

# Types of vectors

Numeric vectors are used for continuous variables

```
y1 <- c(2.1, 3.5, 99.0)
class(y1)

## [1] "numeric"
```

Factors can be used to store categorical variables:

```
y2 <- factor(c("Treatment", "Control", "Treatment"))
y2

## [1] Treatment Control   Treatment
## Levels: Control Treatment
```



# Vectorized arithmetic

How could we calculate the body mass index (BMI = weight/height$^2$) from the following data:

| | Individual | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| Weight | 60 | 72 | 57 | 90 | 95 | 72 |
| Height | 1.8 | 1.8 | 1.7 | 1.9 | 1.7 | 1.9 |

First, create the vectors:

```
weight <- c(60, 72, 57, 90, 95, 72)
height <- c(1.8, 1.8, 1.7, 1.9, 1.7, 1.9)
```

Then, evaluate the equation in just one line:

```
BMI <- weight/height^2
BMI

## [1] 18.51852 22.22222 19.72318 24.93075 32.87197 19.94460
```



# In-class assignment

Calculate the circumference and area of circles with radii: 3,5,6,11

**(1)** Create a new script called "lab1-prob1.R". You can do this by:

- I Clicking on the Console
- II Choosing "File > New Script" from the drop-down menu
- III Clicking on "File > Save as..."

**(2)** Create a vector containing the radii

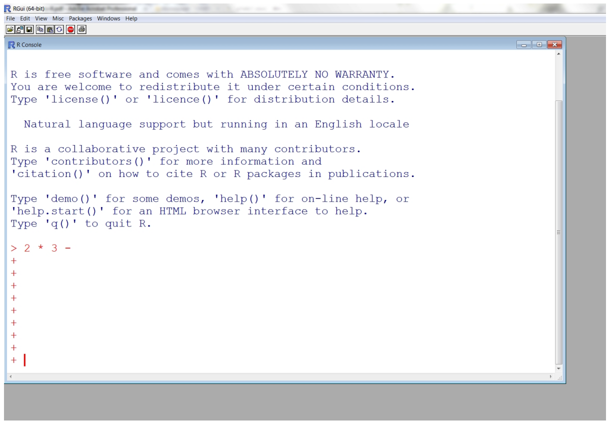
**(3)** Store the computed circumferences and areas in 2 new vectors

**You should be able to do this in just 3 lines in your script. Write all of your code in the script, not in console.**



# A common beginner's problem

If you accidentally hit "return" or fail to complete a command, you will see the cursor on a new line beginning with + instead of >.




Just hit the "Esc" key or complete the command.

# Summarizing vectors

| Number of ticks on 5 dogs | | | | |
|---|---|---|---|---|
| $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ |
| 4 | 7 | 2 | 3 | 150 |

What is the sum of this vector? $\sum_{i=1}^{5} y_i = ???$

```
y <- c(4,7,2,3,150)
sum(y)

## [1] 166
```

What is the mean? $\frac{\sum_{i=1}^{5} y_i}{5} = ???$

```
mean(y)

## [1] 33.2
```

And the variance? $\frac{\sum_{i=1}^{5}(y_i - \bar{y})^2}{5-1} = ???$

```
var(y)

## [1] 4266.7
```



# Indexing vectors

Extract the first and third elements of a vector

```
y <- c(2, 4, 8, 4, 25)
y.sub1 <- y[c(1,3)]
y.sub1

## [1] 2 8
```

Remove the second element

```
y.sub2 <- y[-2]
y.sub2

## [1]  2  8  4 25
```

Rearrange the order of the vector

```
y.re <- y[c(5,4,3,2,1)]
y.re

## [1] 25  4  8  4  2
```



# Indexing vectors

Which elements of the vector are greater than 4? (logical test)

```
y <- c(2, 4, 6, 4, 25)
y>4

## [1] FALSE FALSE  TRUE FALSE  TRUE
```

Extract the elements greater than 4 (logical indexing)

```
y.sub4 <- y[y>4]
y.sub4

## [1]  6 25
```



# Data frames

Most basic datasets are stored as `data.frames`

They are like a matrix in which each column can be a different type of vector (numeric, factor, etc...)

They have attributes for row names (e.g. the names of the experimental units) and column names (e.g. the names of the response and predictor variables)

# Creating a data frame

Simple example with 3 variables measured at 4 sites

```
y <- c(3, 9, 7, 4)
x1 <- factor(c('High', 'High', 'Low', 'Low'))
x2 <- c(2.2, 3.4, 4.4, 3.9)
mydata <- data.frame(Goats=y, Elev=x1, Temp=x2)
rownames(mydata) <- c('Site1', 'Site2', 'Site3', 'Site4')
mydata

##       Goats Elev Temp
## Site1     3 High  2.2
## Site2     9 High  3.4
## Site3     7  Low  4.4
## Site4     4  Low  3.9
```



# Indexing data frames

Bracket method. Extract data from row 1, columns 1 and 3

```
mydata[1,c(1,3)]

##       Goats Temp
## Site1     3  2.2
```

Bracket method. Extract from rows 2 and 3, columns 1 and 3

```
mydata[c('Site2', 'Site3'), c('Goats', 'Temp')]

##       Goats Temp
## Site2     9  3.4
## Site3     7  4.4
```

Dollar sign method. Pull out column 1

```
mydata$Elev

## [1] High High Low  Low
## Levels: High Low
```



# Summarizing data frames

View the data as a ‘spreadsheet’

```
View(mydata)
```

Compute some summary statistics

```
summary(mydata)

##      Goats         Elev        Temp
##  Min.   :3.00   High:2   Min.   :2.200
##  1st Qu.:3.75   Low :2   1st Qu.:3.100
##  Median :5.50            Median :3.650
##  Mean   :5.75            Mean   :3.475
##  3rd Qu.:7.50            3rd Qu.:4.025
##  Max.   :9.00            Max.   :4.400
```

A very compact summary

```
str(mydata)

## 'data.frame': 4 obs. of  3 variables:
##  $ Goats: num  3 9 7 4
##  $ Elev : Factor w/ 2 levels "High","Low": 1 1 2 2
##  $ Temp : num  2.2 3.4 4.4 3.9
```



# Importing and exporting data

`read.csv` and `write.csv` are easy options, but there are many more possibilities that we won’t cover.

Export the `data.frame` we created earlier

```
write.csv(mydata, file="mydata.csv")
```

Confirm that it is there:

```
getwd() # Go to this location and look for 'mydata.csv'
```

Read it back in:

```
mydata2 <- read.csv("mydata.csv")
```

# The working directory

The working directory is the location on your computer where **R** will look for files by default.

You can check your working directory like this:

```
getwd()

## [1] "c:/Work/exp-design/labs/intro-to-R"
```

Change your working directory to another location:

```
## Note the forward slashes, which could be replaced by "\\"
setwd("C:/work/courses/")
```

**At the beginning of every R session, you should use** `setwd` **to set your working directory**



# The workspace

Viewing the objects in your workspace

```
ls()

##  [1] "BMI"      "clean"    "filestub" "height"   "mydata"   "mydata2"
##  [7] "open"     "reqval"   "rnw.file" "tangle"   "weight"   "x1"
## [13] "x2"       "y"        "y.re"     "y.sub1"   "y.sub2"   "y.sub4"
## [19] "y1"       "y2"       "z"
```

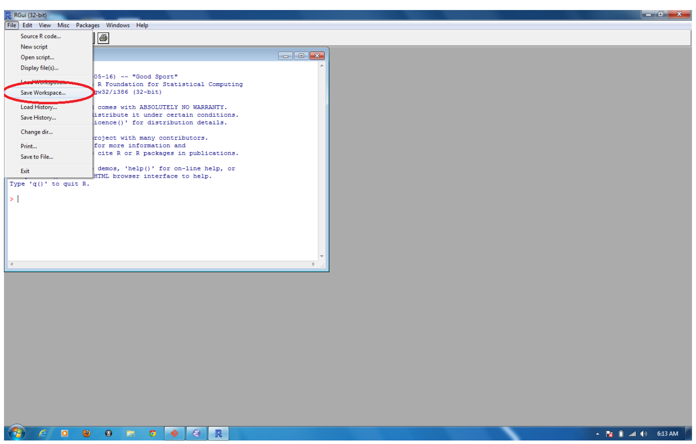
Removing (deleting) some objects

```
rm(x1, x2, mydata2, y, y1, y2, y.re,
   y.sub1, y.sub2, y.sub4, height, weight, BMI, z)
ls()

## [1] "clean"    "filestub" "mydata"   "open"     "reqval"   "rnw.file"
## [7] "tangle"
```



# Saving and restoring the workspace



If you prefer commands over point-and-click:

```
save.image("myimage.RData") # Save all objects to file
load("myimage.RData")       # Load the saved workspace
```



# Additional Resources

'Official' manuals

```
help.start()
```

Useful books

- Venables, W.N. and B.D. Ripley. 2002. Modern Applied Statistics with S, 4th ed. Springer.
- Crawley, M.J.. 2013. The **R** Book, 2nd ed. Wiley

Online

- Always Google your error messages
- `http://stackoverflow.com/`
- `https://preludeinr.com/`

# Assignment – Part I

**(1)** Create an **R** script named something like `Chandler-R_lab1.R`

**(2)** In your **R** script, write code to create a `data.frame` that contains the information shown in the table below

**(3)** Add code to export the `data.frame` as a .csv file and then import it back into **R**.

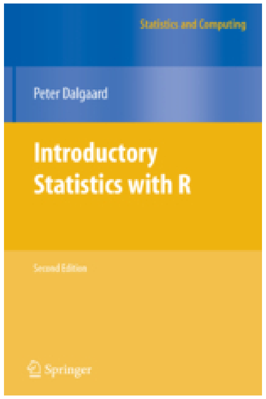
| Individual | Mass | Weight | Treatment |
|---|---|---|---|
| 1 | 3 | 4 | Control |
| 2 | 3 | 5 | Control |
| 3 | 2 | 4 | Control |
| 4 | 4 | 6 | Treatment |
| 5 | 5 | 5 | Treatment |
| 6 | 5 | 7 | Treatment |

Upload your **R** script[1] to ELC before your next lab. The script should be self-contained, meaning that it will run correctly when we copy and paste it in the console.

[1] If you are familiar with RMarkdown, you can submit a `.Rmd` file instead of a `.R` file



# Assignment – Part II




Read chapters 1, 4, & 5 before next lab

Dalgaard, P. 2008. Introductory Statistics with R. 2nd edition. Springer. Available for free through the UGA library:

`http://preproxy.galib.uga.edu/login?url=http://dx.doi.org/10.1007/978-0-387-79054-1`